Gaming on AWS

AWS와 함께한 쿠키런 서버 Re-architecting 사례

홍성진 Senior Technical Lead
데브시스터즈

DEVSISTERS

홍성진 (sungjinhong@devsisters.com)

- 시니어 테크니컬 리드, 데브시스터즈
    - LINE COOKIE RUN, 쿠키런 for Kakao, 오븐브레이크 2 서버 개발
- 시니어 소프트웨어 엔지니어, 위클레이
    - 영상통화 앱 클라이언트 및 서버 개발
    - 프로토타입 서버 개발 (C, Erlang, MySQL, Hadoop, HBase, Riak, Redis)
    - Lucene, Solr와 한글 형태소분석기 관련 작업, 웹 크롤러 및 머신러닝 기법 구현
- 통계기계번역팀, 소프트웨어 엔지니어링 인턴, 구글 코리아
    - 형태소분석, 품사태깅, 웹 크롤러 구현, 훈련용 데이터 크롤링
- Django Project Contributor
    - i18n 관련 기능 구현, 버그픽스, 한글 번역

Agenda

- 오븐브레이크2 서버의 구조
- 기존 서버 관리방식의 문제점
- Maven/CircleCI를 사용한 소스코드 형상관리
- Chef를 사용한 서버 형상관리
- CloudFormation을 사용한 인프라 형상관리
- AutoScale을 사용한 동적 처리용량 확장
- 최근 쿠키런 서버 동향

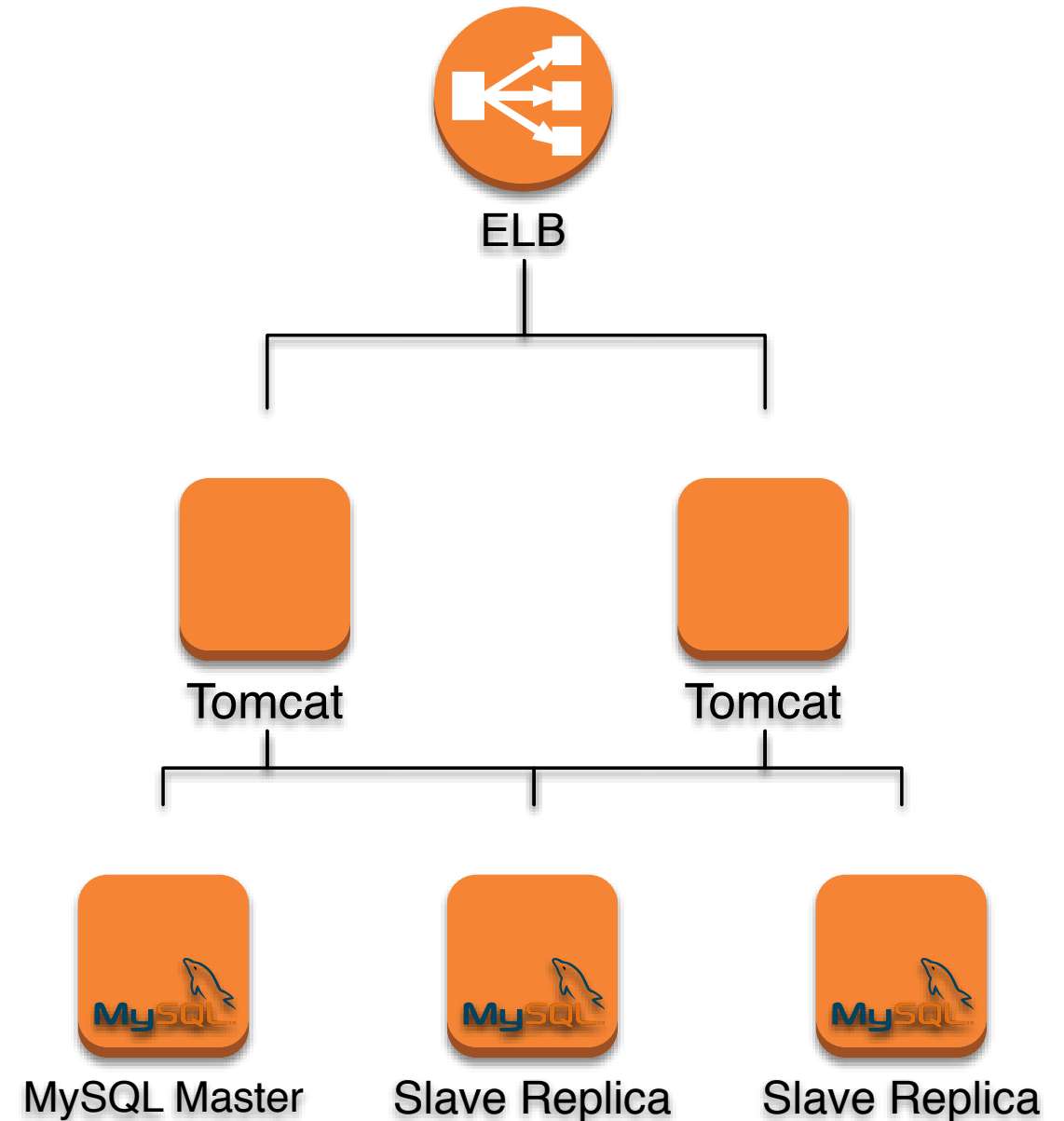
오븐브레이크2

전형적인 서버 구조의 예

오븐브레이크2 서버 구조

- 하드웨어 기반 서버를 본따 만든 서버 구성
- 가용성을 위해 로드밸런서 아래 두 대의 API 서버가 분산되어 있음
- 사용량 증가를 우려해 과도한 스팩을 사용
  - 데이터베이스 - m2.2xlarge
  - API 서버 - m1.large
- 마스터-슬래이브 구조의 데이터베이스
- Backup과 파일 공유를 위한 NFS 서버
- US-East에서 서비스 중

DEVSISTERS

기존 서버 관리 방식의 한계

- 새로운 서버를 만들어야 할 경우 문제
  - 매뉴얼 부족, 있는 자료는 오래된 내용
  - 관리자가 교체되면서 회사에 남아있는 서버지식이 점점 사라짐
  - Tomcat JVM에 설정되어있던 locale 환경설정, 누가 기억하지?
- 사용자 증가에 따른 관리 피로도 증가
  - 급격한 확장에 대응이 불가능하기 때문에 오버스팩 서버 도입
  - 오버스팩 서버 도입에 따른 비용 문제 발생

쿠키런 서버 Re-architecting

AWS 클라우드에 맞는 서버 재설계를 해보자

Maven/CircleCI 소스코드 형상관리

- Maven을 사용하여 Dependency 및 버전 관리
- Github, CircleCI를 사용하여 branch별 continuous integration 및 artifact 관리
- 모든 변경사항은 코드리뷰/테스트 후 master 브렌치로 통합되어 최종 war파일 생성

Branches

Showing 11 branches not merged into master. View merged branches.

master
✓ Last updated 7 days ago by serialx.

Branch	Behind	Ahead
develop-line-release ✗ Last updated 2 hours ago by julingks.	5 behind	22 ahead
update_history ✓ Last updated 4 days ago by julingks.	1 behind	16 ahead
develop ✓ Last updated 5 days ago by julingks.	1 behind	15 ahead
autoblock2 ✓ Last updated 6 days ago by julingks.	1 behind	12 ahead
autoblock ✓ Last updated 6 days ago by julingks.	13 behind	7 ahead

Chef를 사용한 서버 형상관리

- 서버 설정을 JSON과 Ruby를 사용하여 소스코드처럼 Git으로 관리
    - 설정이 소스코드로 문서화 및 형상관리
- 예1: Tomcat을 설치하고 max heap, G1 GC, file.encoding JVM 환경설정
- 예2: 모든 서버에 공통적으로 devsisters 계정을 만들고 htop, dstat, sysstat 설치
- 예3: 서버 운영시 빼먹기 쉬운 여러 설정들을 공통적으로 적용 (ulimit 제한 등)
- <u>언제 어디서든 5분 이내 쿠키런 서버를 원하는만큼 구축할 수 있다</u>

```ruby
directory "/home/devsisters" do
  owner "devsisters"
  group "devsisters"
  mode 0755
  action :create
end

user_ulimit "root" do
  filehandle_limit 65535
end

user_ulimit "devsisters" do
  filehandle_limit 65535
end

package "htop"
package "dstat"
package "sysstat"
```

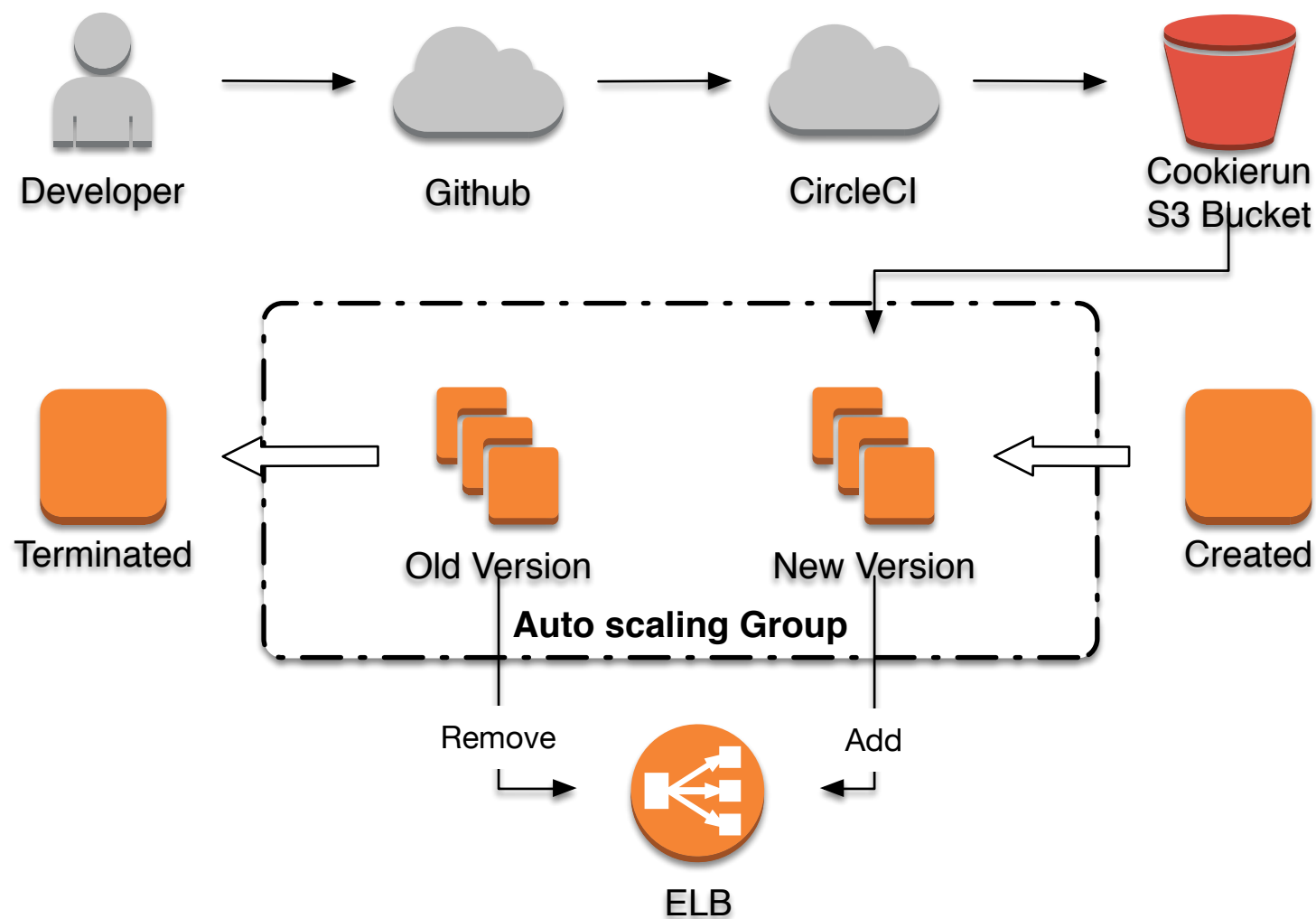
CloudFormation을 사용한 인프라 형상관리

- EC2, ELB, AutoScale 등을 모두 JSON 파일 형태로 템플릿화 하여 관리하는 기능
- 인프라를 소스코드처럼 관리할 수 있음
  - 자동적인 인프라의 문서화 및 형상관리
- 템플릿을 수정하여 AWS에 업로드 하면 기존에 떠있던 클러스터가 수정된다
- <u>언제 어디서든 30분 이내 쿠키런 서버 인프라를 구축할 수 있다</u>

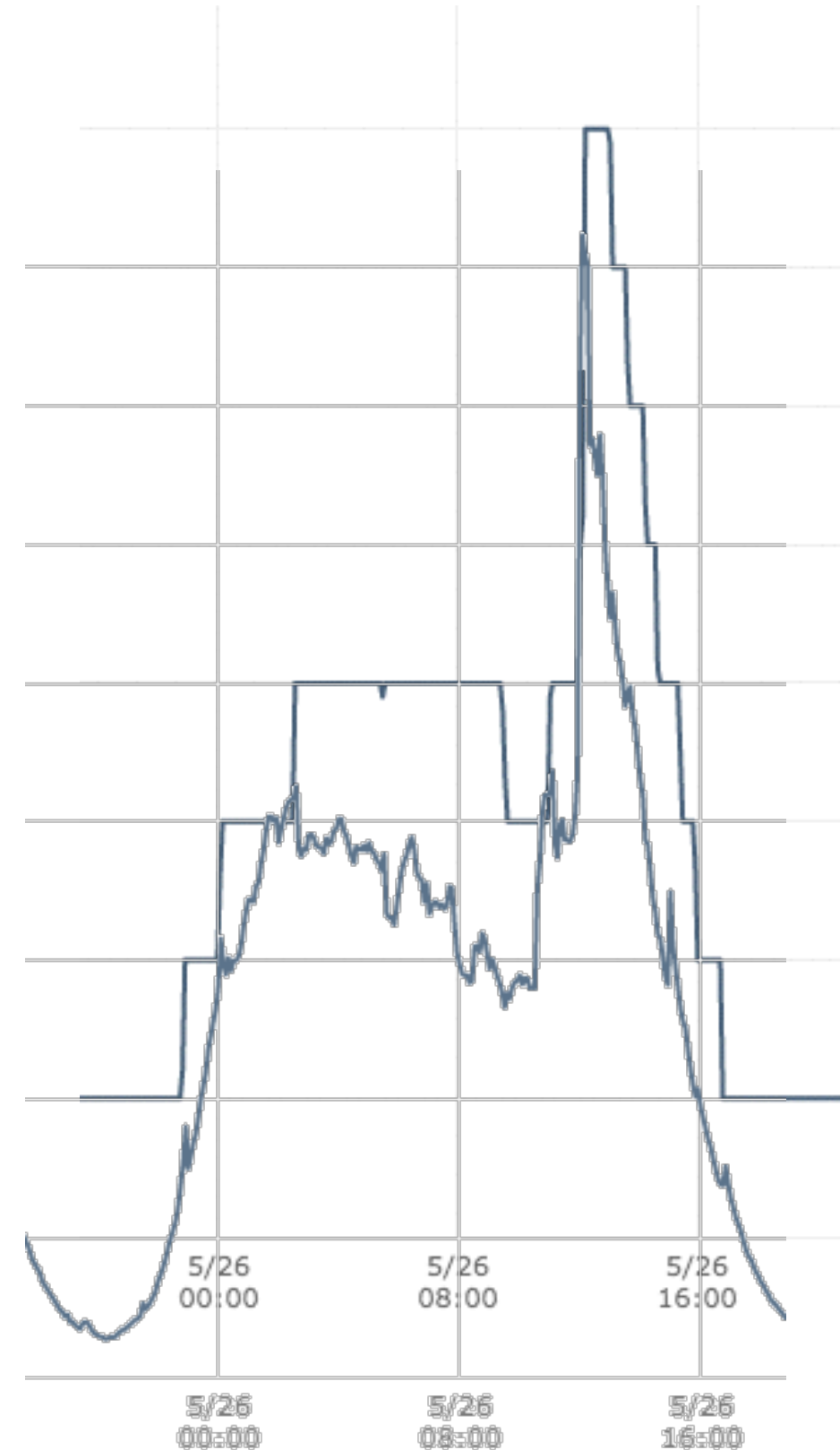
```
"CPUAlarmHigh": {
 "Type": "AWS::CloudWatch::Alarm",
 "Properties": {
    "AlarmDescription": "Scale-up if CPU > 60% for 1 minutes",
    "MetricName": "CPUUtilization",
    "Namespace": "AWS/EC2",
    "Statistic": "Average",
    "Period": "60",
    "EvaluationPeriods": "1",
    "Threshold": "60",
    "AlarmActions": [ { "Ref": "WebServerScaleUpPolicy" } ],
    "Dimensions": [
      {
        "Name": "AutoScalingGroupName",
        "Value": { "Ref": "WebServerGroup" }
      }
    ],
    "ComparisonOperator": "GreaterThanThreshold"
  }
},
```

	Name	Created	Status	Description
☐	cookierun-web-prod	2013-11-18 23:42:41 UTC+0900	UPDATE_COMPLETE	Cookierun Multi-AZ tomcat stack.
☐	cookierun-db-prod	2013-04-01 10:42:33 UTC+0900	CREATE_COMPLETE	Cookierun Multi-AZ RDS stack.

AutoScale을 사용한 동적 처리용량 확장

- Maven으로 만든 war artifact를 S3 Bucket에 업로드
- Chef와 CloudFormation기반으로 만들어진 AutoScaling Group이 Rolling Update 진행
- 5% 정도 진행하고 개발자가 주의깊게 모니터링한 후 100% 패치 작업 진행 (문제 있으면 롤백)

AutoScale 회고

- 장애가 일어난 불량 노드를 알아서 대체
- Scale out 임계치를 잘 설정하는것이 중요
  - 규모가 배가 되었다면 AutoScale 전략도 규모에 맞게 수정하자
  - 인스턴스 크기를 키우는것도 방법
  - m1.medium > m1.large > c1.xlarge
- 크리스마스 연휴 때 c1.xlarge 120대
  - Quota limit에 걸려 문제가 될 뻔
  - Business Support로 빠르게 대처

AutoScale 전략

- 인스턴스 c1.xlarge
  - 8 vCPU, 20 ECU, 7GB Memory
  - c3.2xlarge로 이전 검토 중 - 비슷한 가격에 더 높은 성능과 메모리
- 1분동안 avg CPU > 60% 라면 인스턴스 4대 증가
- 2분동안 max CPU > 80% 라면 인스턴스 4대 증가
  - 급격한 사용자 증가가 있을 경우 빠른 대응을 위한 설정
- 2분동안 avg CPU < 25% 라면 인스턴스 2대 감소
- 최소 4대 및 최대 400대로 설정
  - 의도하지 않은 과도한 과금 방지를 위해서는 최대 수치를 적절하게 조절해야함
- Reserved Instance로 비용감소 검토 중
  - 예) 4대 Heavy Utilization, 10대 Light Utilization

클라우드에 맞는 서버 설계 수정

- 클라우드의 서버는 수명이 짧다. 속칭 하루살이 (ephemeral)
  - Cron Job을 수행하기 위해 한 서버가 마스터로 설정되어야하는 필요성 제거
    - ✓ Shared-nothing 구조로 변경
  - Game Log를 손쉽게 Log Server로 Shipping할 수 있도록 Logging 구조 개선
    - ✓ slf4j/Logback기반으로 변경, Logstash agent등의 도입
- 완벽한 Consistency를 보장하는 RDB에서 적당한 Consistency를 추구하는 NoSQL로
  - 약간의 Consistency 희생으로 수평적확장성, 재해복구, 관리의 용이성 확보
  - 적당한 Consistency 에서도 서버 로직이 작동하도록 구현 개선

Re-architecting 결과

AWS

Internet

Availability Zone-1

Availability Zone-2

EC2

EC2

EC2

EC2

EC2

EC2

EC2

EC2

M R R

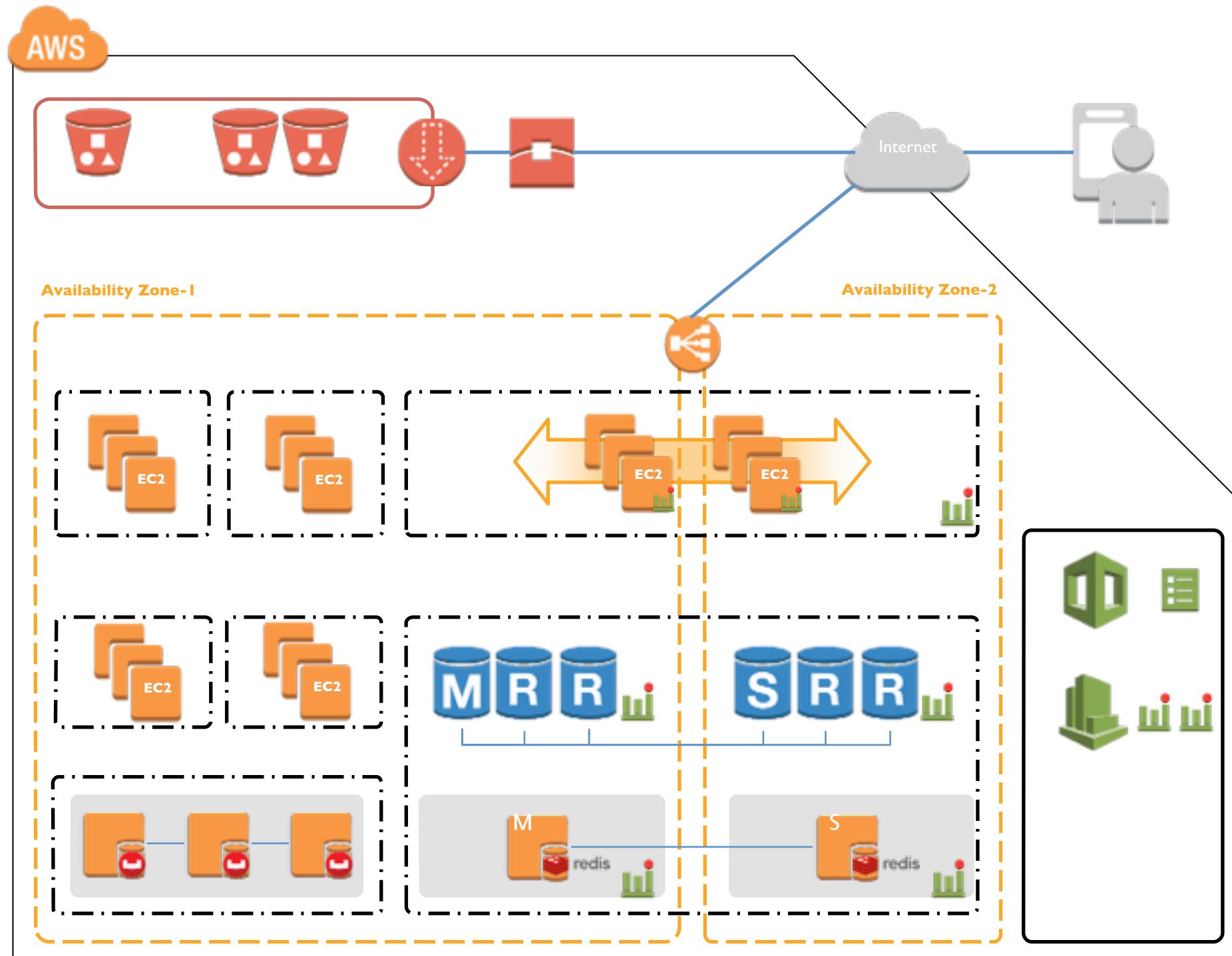
S R R

M

redis

S

redis

최근 쿠키런 서버 동향 (1/2)

- 새로운 인스턴스 유형 검토 중
  - c1.xlarge 대신 c3.2xlarge 인스턴스 사용
  - hi1.4xlarge 대신 cr1.8xlarge 인스턴스 사용
  - 모두 성능과 가격면에서 합격점, 인기가 많아 사용할 수 있는지가 관건
- NoSQL Couchbase 시범적 도입 성공
  - 메인 사용자 데이터를 Couchbase 기반으로 마이그레이션 진행함
  - 성능과 수평 확장성, 그리고 관리 편의성에서 모두 높은 점수
  - cr1.8xlarge 4대, 총 976GB Memory, 960GB SSD Storage 클러스터 운영 중
  - Reserved Instance를 사용하여 비용절감 중

최근 쿠키런 서버 동향 (2/2)

- 최근 눈에 띄는 Seoul - Tokyo Region Latency 감소
- 밤, 낮, 주말과 관계 없이 40ms의 고정된 Latency를 지속적으로 유지
- 한국 커뮤니티 내에서도 이러한 현상을 확인
- AWS측에서 국내 ISP와 협력을 진행한게 아닌지 추측

Primary DNS - Ping Latency

milliseconds

200
150
100
50
0

Week 05 Week 06 Week 07 Week 08

RRDTOOL / TOBI OETIKER

Ping Latency to Primary DNS Current: 40.32 Average: 92.03 Maximum:

Q&A

감사합니다

홍성진 <sungjinhong@devsisters.com>